In [3]:
```
#Ans1:
import regex as re
Sample_Text = 'Python Exercises, PHP exercises.'
re.sub(r"\W",":",Sample_Text)
```

Out[3]:
```
'Python:Exercises::PHP:exercises:'
```

In [19]:
```
#Ans2:
import pandas as pd
import regex as re
dictionary = {'SUMMARY' : ['hello, world!', 'XXXXX test', '123four, five:; six...']}
data_frame = pd.DataFrame(dictionary)
def remove_text(text):
    remove_text = re.sub(r"[^a-z\s]","",text)
    remove_text = re.sub(r"\s+"," ",remove_text).strip()
    return remove_text
data_frame["SUMMARY"] = data_frame["SUMMARY"].apply(remove_text)
print(data_frame)
```

```
         SUMMARY
0    hello world
1           test
2  four five six
```

In [24]:
```
#Ans3:
import regex as re
str1 = "qwerty uiop asdfgh jkl"
string_pattern = (r"\w{4,}")
regex_pattern = re.compile(string_pattern)
result = regex_pattern.findall(str1)
print(result)
```

```
['qwerty', 'uiop', 'asdfgh']
```

In [50]:
```
#Ans4:
import re
str1 = "qwerty uiop asdfgh jkl zxcvbnm"
string_pattern = (r"\w{3,5}")
regex_pattern = re.compile(string_pattern)
result = regex_pattern.findall(str1)
print(result)
```

```
['qwert', 'uiop', 'asdfg', 'jkl', 'zxcvb']
```

In [12]:
```
#Ans5:
import re
def remove_brackets(strings):
    pattern = re.compile(r'\([^)]*\)')
    result = [pattern.sub('', s) for s in strings]
    return result
sample_text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Sci
output = remove_brackets(sample_text)
for item in output:
    print(item)
```

```
example
hr@fliprobo
github
Hello
Data
```

In [13]:
```
#Ans6:
import re
```

```
def remove_brackets(strings):
    pattern = re.compile(r'\([^)]*\)')
    result = [pattern.sub('', s) for s in strings]
    return result
sample_text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Sci
output = remove_brackets(sample_text)
for item in output:
    print(item)
```

```
example
hr@fliprobo
github
Hello
Data
```

In [15]:
```
#Ans7:
import regex as re
sample_text = "ImportanceOfRegularExpressionsInPython"
result = re.findall(r'[A-Z][a-z]*', sample_text)
print(result)
```

```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [24]:
```
#Ans8:
import re
def insert_spaces(text):
  pattern = r'(\d+)([A-Za-z]+)'
  result = re.sub(pattern, r'\1 \2', text)
  return result
text = "RegularExpression1IsAn2ImportantTopic3InPython"
output = insert_spaces(text)
print(output)
```

```
RegularExpression1 IsAn2 ImportantTopic3 InPython
```

In [25]:
```
#Ans9:
import re
def organising_spaces(text):
    modified_text = re.sub(r'([A-Z0-9][a-z]*)', r' \1', text)
    return modified_text.strip()
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
output = insert_spaces(sample_text)
print(output)
```

```
Regular Expression 1 Is An 2 Important Topic 3 In Python
```

In [27]:
```
#Ans10:
import regex as re
import pandas as pd
url = "https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_score_data
df = pd.read_csv(url)
df['first_five_letters'] = df['Country'].str[:6]
print(df)
```

```
         Country                          Region  Happiness Rank  \
0    Switzerland                  Western Europe               1
1        Iceland                  Western Europe               2
2        Denmark                  Western Europe               3
3         Norway                  Western Europe               4
4         Canada                   North America               5
..           ...                             ...             ...
153       Rwanda              Sub-Saharan Africa             154
154        Benin              Sub-Saharan Africa             155
155        Syria  Middle East and Northern Africa             156
156      Burundi              Sub-Saharan Africa             157
157         Togo              Sub-Saharan Africa             158

     Happiness Score  Standard Error  Economy (GDP per Capita)   Family  \
0              7.587         0.03411                   1.39651  1.34951
1              7.561         0.04884                   1.30232  1.40223
2              7.527         0.03328                   1.32548  1.36058
3              7.522         0.03880                   1.45900  1.33095
4              7.427         0.03553                   1.32629  1.32261
..               ...             ...                       ...      ...
153            3.465         0.03464                   0.22208  0.77370
154            3.340         0.03656                   0.28665  0.35386
155            3.006         0.05015                   0.66320  0.47489
156            2.905         0.08658                   0.01530  0.41587
157            2.839         0.06727                   0.20868  0.13995

     Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
0                     0.94143  0.66557                        0.41978
1                     0.94784  0.62877                        0.14145
2                     0.87464  0.64938                        0.48357
3                     0.88521  0.66973                        0.36503
4                     0.90563  0.63297                        0.32957
..                        ...      ...                            ...
153                   0.42864  0.59201                        0.55191
154                   0.31910  0.48450                        0.08010
155                   0.72193  0.15684                        0.18906
156                   0.22396  0.11850                        0.10062
157                   0.28443  0.36453                        0.10731

     Generosity  Dystopia Residual first_five_letters
0       0.29678            2.51738             Switze
1       0.43630            2.70201             Icelan
2       0.34139            2.49204             Denmar
3       0.34699            2.46531             Norway
4       0.45811            2.45176             Canada
..          ...                ...                ...
153     0.22628            0.67042             Rwanda
154     0.18260            1.63328              Benin
155     0.47179            0.32858              Syria
156     0.19727            1.83302             Burund
157     0.16681            1.56726               Togo

[158 rows x 13 columns]
```

In [29]:

```python
#Ans11:
import regex as re
def match_txt(text):
        pattern = '^[a-zA-Z0-9_]*$'
        if re.search(pattern,  text):
                return ('match found')
        else:
                return('match not found')
```

In [31]:
```python
#Ans12:
import regex as re
def num_matching(string):
    sample_text = re.compile(r"^7")
    if sample_text.match(string):
        return True
    else:
        return False
print(num_matching('123456'))
print(num_matching('7890'))
```

```
False
True
```

In [34]:
```python
#Ans13:
import regex as re
ip = "112.115.05.198"
regex_pattern = re.sub('\.[0]*', '.', ip)
print(regex_pattern)
```

```
112.115.5.198
```

In [42]:
```python
#Ans14:
import regex as re
sample_text = "On August 15th 1947 that India was declared independent from British colo
pattern = r"\b([A-Z][a-z]+ \d{1,2}(?:st|nd|rd|th)? \d{4})\b"
match = re.findall(pattern, sample_text)
date_string = match[0] if match else None
print(date_string)
```

```
August 15th 1947
```

In [43]:
```python
#Ans15:
import regex as re
searched_words = ["fox","dog","horse"]
sample_text = 'The quick brown fox jumps over the lazy dog.'
for word in searched_words:
    if re.search(word,sample_text):
        print("match found")
    else:
        print("no match")
```

```
match found
match found
no match
```

In [47]:
```python
#Ans16:
import regex as re
pattern = 'fox'
sample_text = 'The quick brown fox jumps over the lazy dog.'
match = re.search(pattern, sample_text)
s = match.start()
e = match.end()
print('Matched "%s" in "%s" from %d to %d ' % \
    (match.re.pattern, match.string, s, e))
```

```
Matched "fox" in "The quick brown fox jumps over the lazy dog." from 16 to 19
```

In [49]:
```python
#Ans17:
import regex as re
sample_text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
```

```
for match in re.findall(pattern, sample_text):
    print(match)
```

```
exercises
exercises
exercises
```

In [53]:
```
#Ans18:
import regex as re
sample_txt = "Data Science is fasinating."
pattern = "fasinating"
for match in re.finditer(pattern,sample_txt):
    s = match.start()
    e = match.end()
print(match)
```

```
<regex.Match object; span=(16, 26), match='fasinating'>
```

In [57]:
```
#Ans19:
import regex as re
def change_format(dt):
        return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)
sample_date = "1999-08-30"
print("Date YYY-MM-DD Format:",sample_date)
print("Date DD-MM-YYYY Format:",change_format(sample_date))
```

```
Date YYY-MM-DD Format: 1999-08-30
Date DD-MM-YYYY Format: 30-08-1999
```

In [62]:
```
#Ans20:
import regex as re

def all_decimal_numbers(string):
  pattern = re.compile(r'\d+\.\d{1,2}')
  decimal_numbers = re.findall(pattern, string)
  return decimal_numbers
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
result = all_decimal_numbers(sample_text)
print(result)
```

```
['01.12', '0132.12', '2.31', '145.8', '3.01', '27.25', '0.25']
```

In [1]:
```
#Ans21:
import regex as re
sample_text = "There are total number of 11 players in a football team."
for m in re.finditer("\d+",sample_text):
    print(m.group(0))
    print("index pos:",m.start())
```

```
11
index pos: 26
```

In [2]:
```
#Ans22:
import regex as re
sample_text = "My marks in each semester are: 947, 896, 926, 524, 734, 950, 642"
numeric_value = re.findall(r"\d+",sample_text)
if numeric_value:
    maximum_value = max(map(int,numeric_value))
    print(maximum_value)
else:
    print("no numeric value")
```

```
950
```

In [10]:
```
#Ans24:
import re
```

```
def match_txt(text):
        pattern = '[A-Z]+[a-z]+$'
        if re.search(pattern, text):
                return "match found"
        else:
                return("match not found")
print(match_txt("Data Science"))
print(match_txt("data science"))
```

```
match found
match not found
```

In [7]:
```
#Ans23:
import regex as re

def insert_spaces(text):
  pattern = r'([A-Z][a-z]+)'
  result = re.sub(pattern, r' \1', text)
  result = result.strip()
  return result
sample_text = "RegularExpressionIsAnImportantTopicInPython"
output = insert_spaces(sample_text)
print(output)
```

```
Regular Expression Is An Important Topic In Python
```

In [13]:
```
#Ans25:
import regex as re
def remove_dupes(sentence):
    pattern = (r"\b(\w+)\s+\1\b")
    result = re.sub(pattern, r'\1', sentence)
    return result
sample_text = "Hello hello world world"
output = remove_dupes(sample_text)
print(output)
```

```
Hello hello world
```

In [14]:
```
#Ans26:
import regex as re
def check_string(string):
  pattern = r"\w$"
  match = re.search(pattern, string)
  if match:
    return True
  else:
        return False
```

In [16]:
```
#Ans27:
import regex as re
sample_text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization
pattern = r'#\w+'
hashtag_extraction = re.findall(pattern, sample_text)
print(hashtag_extraction)
```

```
['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [18]:
```
#Ans28:
import regex as re
sample_text = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Th
pattern = r'<U\+[0-9A-Fa-f]+>'
output_text = re.sub(pattern, '', sample_text)
print(output_text)
```

```
@Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization are all
different party leaders
```

In [20]:
```
#Ans29:
import regex as re
regex_pattern = (r"\d{2}-\d{2}-\d{4}")
sample_text = "Ron was born on 12-09-1992 and he was admitted to school 15-12-1999."
dates = re.findall(regex_pattern,sample_text)
print(dates)
```

```
['12-09-1992', '15-12-1999']
```

In [ ]:
```

```

In [27]:
```
#Ans30:
import regex as re
def remove_words(string):
  pattern = re.compile(r'\b\w{2,4}\b')
  modified_string = re.sub(pattern, '', string)
  return modified_string
sample_text = "The following example creates an ArrayList with a capacity of 50 elements
expected_output = "following example creates ArrayList a capacity elements. 4 elements a
result = remove_words(sample_text)
print(result)
```

```
 following example creates  ArrayList  a capacity   elements. 4 elements   added   Array
List   ArrayList  trimmed accordingly.
```

In [ ]:
```

```